UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I
CENTRO CIÊNCIA E TECNOLOGIA
CURSO DE CIENCIA DA COMPUTAÇÃO

LUCAS BARBOSA OLIVEIRA

DESENVOLVIMENTO DE UM AGREGADOR DE DADOS PARA DISPOSITIVOS MÉDICOS PESSOAIS

CAMPINA GRANDE - PB
2017

LUCAS BARBOSA OLIVEIRA

DESENVOLVIMENTO DE UM AGREGADOR DE DADOS PARA DISPOSITIVOS MÉDICOS PESSOAIS

Trabalho de Conclusão de Curso em Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Computação.

Área de concentração: Sistemas Distribuídos.

Orientador: Prof. Dr. Paulo Eduardo e Silva Barbosa.

CAMPINA GRANDE - PB
2017

O48d Oliveira, Lucas Barbosa.
Desenvolvimento de um agregador de dados para dispositivos médicos pessoais [manuscrito] / Lucas Barbosa Oliveira. - 2017.
49 p. : il. color.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2017.
"Orientação: Prof. Dr. Paulo Eduardo e Silva Barbosa, Departamento de Computação".

1. Dispositivos médicos pessoais. 2. Interoperabilidade. 3. Conectividade. 4. ISO IEEE 11073. I. Título.

21. ed. CDD 004.6

LUCAS BARBOSA OLIVEIRA

DESENVOLVIMENTO DE UM AGREGADOR DE DADOS PARA DISPOSITIVOS MÉDICOS PESSOAIS

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito à obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 11 de Agosto de 2017.

Prof. Dr. Paulo Eduardo e Silva Barbosa (UEPB)
Orientador(a)

Prof. Dr. Sabrina Souto (UEPB)
Examinador(a)

Prof. Fernando Matos (UFPB)
Examinador(a)

A Deus, pela dedicação, apoio, companheirismo e amizade, DEDICO.

AGRADECIMENTOS

Primeiramente louvo a Deus por me agraciar com o dom da vida e me possibilitar esta conquista, consciente de que tudo está sob o seu decreto.

Em segundo lugar, agradeço a honrosa Universidade Estadual da Paraíba por me conceder o privilégio de terminar a graduação no curso de Ciência da Computação e haver disponibilizado toda a estrutura necessária, juntamente com o seu corpo de docentes, entre estes destaco a grande relevância do meu professor e orientador Paulo Eduardo e Silva Barbosa por ter me instruído e fundamentado em toda a minha vida acadêmica. Agradeço ainda aos diversos colegas que estudaram comigo durante esse período (os quais se fossem todos mencionados aqui, certamente não caberiam), pois as amizades construídas ali se perpetuarão ao longo da minha vida. Gratidão, em especial, a todo o laboratório do NUTES (a minha segunda casa) e aos amigos Adalberto Monteiro, Alisson Santos, Leonardo Camara e Yasser Nascimento.

Por último, mas não menos importantes, agradeço a toda a minha família, minha mãe, Gildênia Barbosa Oliveira, meu pai, Alírio de Souza Oliveira e irmãos, Saulo Barbosa Oliveira, Samuel Barbosa Oliveira e Aline Barbosa Oliveira. Estes foram os alicerces postos por Deus e tiveram grande relevância em todos os aspectos da minha vida.

Assim como comecei os agradecimentos, eu concluo louvando a Deus, tendo total certeza de que se o tudo que fazemos nessa vida não visa à glória de Dele, o tudo se resume ao nada.

RESUMO

Nos dias atuais, o avanço da tecnologia gera cada vez mais comunicação e acesso à informação para a sociedade, resultando num maior oferecimento de serviços aos usuários. A isto é dado o nome de “Revolução dos Produtos”, onde a partir da utilização de softwares avançados, os produtos ganham sistemas mais complexos e com inúmeras formas de conectividade, principalmente a partir dos grandes benefícios gerados pela conectividade sem fio. Os dispositivos médicos pessoais estão inseridos neste contexto, pois dia após dia estão se disseminando pelas mais diversas áreas do cotidiano das pessoas, proporcionando, assim, uma melhor qualidade de vida através do monitoramento contínuo da saúde dos mesmos. Em decorrência disso, os mais diversos tipos dispositivos estão sendo criados, tais como, oxímetro, balança, aferidor de pressão, entre outros. Entretanto, devido à fabricação destes dispositivos alguns problemas surgiram, inviabilizando a conectividade e interoperabilidade destes dispositivos com outros sistemas, por exemplo, concentradores de dados e prontuários eletrônicos médicos. Em decorrência disso, este trabalho de conclusão de curso destina-se a demonstrar uma solução para o problema utilizando o antidote, que é uma implementação da Norma IEEE 11073-60201 com o intuito de preencher esta lacuna. Durante o desenvolvimento desta solução, foi criado um Hub utilizando uma plataforma embarcada, dentro da qual foi realizada as devidas configurações, juntamente com diversos outros componentes que de forma colaborativa constituem a proposta deste trabalho, ou seja, a criação de um agregador de dados para dispositivos médicos pessoais. Este agregador de dados estará apto a estabelecer uma comunicação com dispositivos médicos pessoais através do protocolo de comunicação Bluetooth. Para isso, operações como pesquisa, pareamento, despareamento e transferência de dados devem estar disponíveis para o manuseio de um provável usuário final. Para o desenvolvimento desse projeto foram utilizadas diversas ferramentas tecnológicas responsáveis justamente pelo desenvolvimento e comunicação dos componentes, algumas destas são D-Bus, que é um mecanismo de comunicação entre processos, Yocto Project, para construção da imagem do sistema operacional personalizado, e o Node.js que foi utilizada como plataforma de execução da principal linguagem de programação utilizada durante a implementação do projeto, o JavaScript. Além disso, no decorrer do projeto foram desenvolvidos artefatos relacionados aos requisitos do sistema com o intuito de proporcionar uma compreensão mais ampla, como por exemplo, um documento contendo as estórias de usuário, a criação de diagramas de contexto e o desenvolvimento dos principais casos de uso. Por fim, foram utilizados dois dispositivos médicos pessoais que foram fabricados mediante a sua respectiva ramificação da norma IEEE 11073 como demonstração da validação do projeto; tais dispositivos são credenciados pela *Contínua Health Alliance* que é a entidade certificadora dos dispositivos médicos pessoais submetidos a esta norma.

Palavras-Chave: Interoperabilidade, Conectividade, IEEE 11073, D-Bus, Yocto Project, JavaScript, Contínua Health Allience.

ABSTRACT

Nowadays, the advancement of technology generates more and more communication and access to information for the society, resulting in a greater offer of services to users. We call this process "product revolution", where from the use of advanced software, the products gain more complex systems and innumerable forms of connectivity, mainly from the great benefits generated by wireless connectivity. The personal medical devices are inserted in this context, because day after day are spreading through the most diverse areas of people's daily life, thus providing a better quality of life through the continuous monitoring of their health. As a result, the most diverse types of devices are being created, such as, oximeter, balance, pressure gauge, among others. However, due to the manufacture of these devices some problems have arisen, making the connectivity and interoperability of these devices with other systems, for example, data concentrators and electronic medical records impossible. As a result, this course completion work is intended to demonstrate a solution to the problem using antidote, which is an implementation of the Standard IEEE 11073-60201 in order to fill this gap. During the development of this solution, a Hub was created using an embedded platform, in which the appropriate configurations were made, along with several other components that collaboratively constitute the proposal of this work, the creation of a data aggregator for Personal medical devices. This data aggregator will be able to establish communication with personal medical devices through the Bluetooth communication protocol. For this, operations such as search, pairing, unpaired and data transfer must be available for the handling of a probable end user. For the development of this project, a number of technological tools were used precisely for the development and communication of the components, some of them are D-Bus, which is a mechanism of communication between processes, Yocto Project, to build the image of the customized operating system, and the Node.js which was used as the execution platform of the main programming language used during the implementation of the project, JavaScript. In addition, during the course of the project, artifacts related to system requirements were developed to provide a broader understanding, such as a document containing user stories, creating context diagrams, and developing key case studies. use. Finally, two personal medical devices were used that were manufactured by their respective branch of the IEEE 11073 standard as demonstration of the validation of the project; Such devices are accredited by the Continuous Health Alliance which is the certifier of the personal medical devices subject to this standard.

Keywords: Interoperability, Connectivity, IEEE 11073, D-Bus, Yocto Project, JavaScript, Continuous Health Allience.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DIM	Domain Information Model
EMR	Electronic Medical Records
HAM	Health Aggregator Manager
HDP	Bluetooth Health Device Profile
IEEE	Institute of Electrical and Electronics Engineers
IoMT	Internet of Medical Things
IoT	Internet of Things
ISO	International Organization for Standardization
JNI	Java Native Interface
MDS	Medical Devices System
MPI	Message Passing Interface
NPM	Node Package Manager
NUTES	Núcleo de Tecnologias Estratégicas em Saúde
SSP	Serial Port Profile
UEPB	Universidade Estadual da Paraíba
UI	User Interface
IPC	Inter-Process Communication

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Com os avanços tecnológicos, diversos movimentos que visam estabelecer a comunicação entre partes de subsistemas vêm sendo evidenciados. Um desses é o movimento relacionado à Internet das coisas *(IoT, sigla em inglês)*, o qual abrange as mais diversas áreas com o intuito de fornecer serviços para a disseminação de informação. Dentro do campo médico, este movimento é conhecido como *Internet of Medical Things (IoMT),* e está muito relacionado as atividades desempenhas por sistemas *HealthCare*, os quais podem ser definidos como uma coleção de dispositivos médicos e aplicações que se conectam a sistemas de TI de saúde através da redes de computadores disponíveis. Com isso, os dispositivos médicos equipados com Wi-Fi podem ser conectados a plataformas nas nuvens, tais como Amazon Web Services, nos quais os dados capturados podem ser armazenados e analisados (ROUSE, 2015).

Em decorrência disso, os dispositivos médicos estão ganhando cada vez mais espaço no cotidiano das pessoas, fatores que podem estar influenciando essa tendência são:

- A constante necessidade de monitoramento que as pessoas devem estar submetidas durante tratamento de alguma patologia;
- A busca pelo aumento da qualidade de vida, uma vez que os pacientes poderiam está sendo monitorados de uma forma periódica;
- A inversão da pirâmide hereditária, gerando assim uma população mais velha e, consequentemente, que necessita de maior comodidade durante algum procedimento médico;

Por esses motivos, têm-se dado grande ênfase nas tecnologias que usam comunicação wireless, tais como, Bluetooth, Wi-fi, InfraRed, ZifBeer RFIC ou NFC com o intuito de proporcionar, como anteriormente mencionado uma maior comodidade, por se acreditar que tais tecnologias possibilitam uma maior flexibilidade ao realizar o monitoramento remoto dos dispositivos. É com isso em mente que diversas empresas de saúde estão se voltando para o desenvolvimento de dispositivos médicos pessoais com estas especificações.

Como consequência dessa enorme quantidade de fabricação de dispositivos, passou-se a existir uma grande heterogeneidade entre tais dispositivos, impossibilitando a interoperabilidade entre outros dispositivos e, até mesmo, com sistemas Electronic Medical Records (EMR). Toda essa heterogeneidade ou falta de comunicação é justificada pelo fato dos dispositivos médicos terem sido desenvolvidos através de protocolos privados de uma determinada empresa.

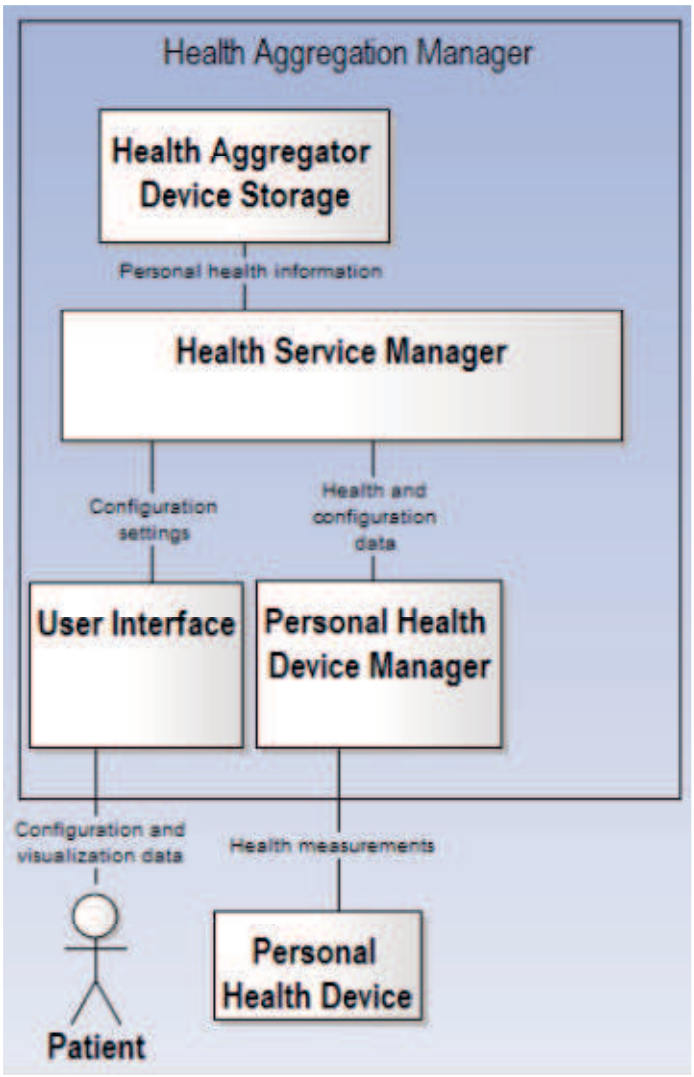
A partir do instante em que se é possível estabelecer a comunicação entre os tais dispositivos e sistemas, diversos benefícios emergem por transitividade. Para ilustrar tais benefícios analisemos os três seguintes casos de uso (FARIA, 2015):

- Primeiro Caso – Atualmente os profissionais médicos realizam plantões em diferentes hospitais espalhados por uma cidade, ou até mesmo fora dela. O impacto da implantação de um agregador de dados pode estar relacionado a um cenário como este, uma vez que, poderia funcionar como um intermediador para o monitoramento remoto dos dispositivos médicos, juntamente com os sinais vitais dos pacientes.
- Segundo Caso – Na perspectiva de um paciente como, por exemplo, uma pessoa em seus exames de rotina ou um atleta que está habituado a realizar exames relativamente simples periodicamente, essa tecnologia proporcionaria um maior conforto, pois ao invés de ter que se direcionar a uma clínica médica com o objetivo de realizar algum exame, ambos agora possuiriam um dispositivo capaz de notificar o médico e até mesmo seus familiares a respeito das suas condições de saúde.
- Terceiro Caso – Dentro dos hospitais existe um grande fluxo de pessoas que necessitam ser examinadas e, por consequência disso, são geradas diversas filas para o atendimento e para realização de uma simples avaliação. Uma possível solução para tal problema seria centralizar os dados obtidos dos dispositivos médicos de forma simultânea, característica essa do agregador de dados de dispositivos médicos pessoais.

1.2 Objetivos

Esse trabalho de conclusão de curso visa estabelecer a interoperabilidade entre dispositivos médicos. Para cumprir com tal objetivo será utilizada uma implementação da norma ISO/IEEE 11073, que, em termos gerais, trata da padronização na fabricação de dispositivos médicos pessoais. A partir dessa implementação será criado um agregador de dados capaz de se comunicar com os mesmos em uma rede de comunicação local (ASARE, VATTAM, *et al.*, 2012). A imagem a seguir demonstra o diagrama de contexto da aplicação que funcionará basicamente como um middleware:

Figura 1 – Diagrama de Contexto do HAM



Fonte: Elaborada pelo autor

O diagrama de contexto do Health Aggregator Manager (isto é o agregador de dados médicos) juntamente com o cenário de uso, consiste basicamente de dois atores e quatro componentes, os quais trabalham de forma colaborativa em prol de estabelecer a comunicação com os dispositivos.

O primeiro ator do sistema é o *Patient*. Esse é o responsável por interagir com o agregador de dados e realizar as operações de pesquisa, pareamento e despareamento do Bluetooth e, até mesmo, realizar uma visualização dos dados recebidos do dispositivo médico.

O segundo ator do sistema é o *Personal Health Device*, ou os dispositivos médicos pessoais é visto basicamente como o produtor dos dados, afinal são os dispositivos que irão gerar as informações que serão emitidas para o agregador de dados posteriormente.

O módulo *User Interface* é responsável por gerenciar as ações de entrada de um possível paciente, médico ou enfermeiro no agregador de dados e permitir que o mesmo possa receber e visualizar os dados enviados dos dispositivos médicos.

O *Personal Health Device Manager* tem como finalidade gerir as operações para estabelecer a conexão com o dispositivo médico, isto inclui as operações de Bluetooth anteriormente mencionadas.

O terceiro componente incluso no agregador de dados é o Health Aggregator Device Storage, que destina-se a armazenar os dados recebidos a partir dos dispositivos médicos. Um grande benefício decorrente deste componente é que a partir dele é possível realizar operações de forma offline e, posteriormente, fazer o envio e a sincronização das informações obtidas para camadas superiores, tal como um Web Service na Amazon ou um framework de coordenação de dispositivos médicos.

O quarto e último componente deve ser compreendido como núcleo do sistema, uma vez que é neste onde residem as operações responsáveis por gerenciar a comunicação com os dispositivos médicos e onde serão implantados os padrões que irão possibilitar a interoperabilidade.

1.3 Estrutura do Trabalho de Conclusão de Curso

Com o intuito de proporcionar uma plena compreensão da proposta apresentada, este trabalho foi organizado obedecendo a uma estrutura ramificada nas seguintes diretrizes:

2. Fundamentação Teórica – Serão apresentados os principais pontos que fundamentaram e induziram o desenvolvimento deste trabalho, juntamente com os conceitos mais relevantes que proporcionarão ao leitor um pleno entendimento deste trabalho de conclusão de curso como um todo.
3. Metodologia - Nesta etapa serão esclarecidas as fases do processo de desenvolvimento da monografia, trazendo consigo um breve resumo das ramificações inerentes à esta seção. Este capítulo contará apenas com duas subseções principais:
   3.1. Ferramentas Tecnológicas – A primeira seção tratará das ferramentas fundamentais para o desenvolvimento deste trabalho de forma breve, sendo abordadas de forma geral as principais peculiaridades das tecnologias.
   3.2. Processo de Desenvolvimento – A segunda ramificação destina-se a mostrar como foram utilizadas tais tecnologias para o desenvolvimento e concretização dos objetivos apresentados anteriormente.
4. Conclusões – Neste capítulo será apresentada uma argumentação em torno do trabalho desenvolvido, com o intuito de recapitular o estado da arte anterior a este trabalho analisando a contribuição científica gerada, e por fim expor futuras linhas de pesquisas derivadas deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 BlueZ

O Bluetooth é uma tecnologia de transmissão de dados sem fio, que se propõe a estabelecer a comunicação entre smartphones, tablets, notebooks, fones de ouvido e os mais diversos periféricos existentes na atualidade. Essa tecnologia foi projetada para atender requisitos de baixo consumo de energia e com baixo alcance, este último variando em torno de três classes que dependem da potência descrita na especificação, estas são (LIMA e GONÇALVES, 2009):

Tabela 1 - Classes de Dispositivos Bluetooth

Classe	**Potência Máxima**	**Faixa de Operação**
Classe 1	100mW (20dBm)	~100 metros
Classe 2	2.5mW (4dBm)	~10 metros
Classe 3	1mW (0dBm)	~1 metros

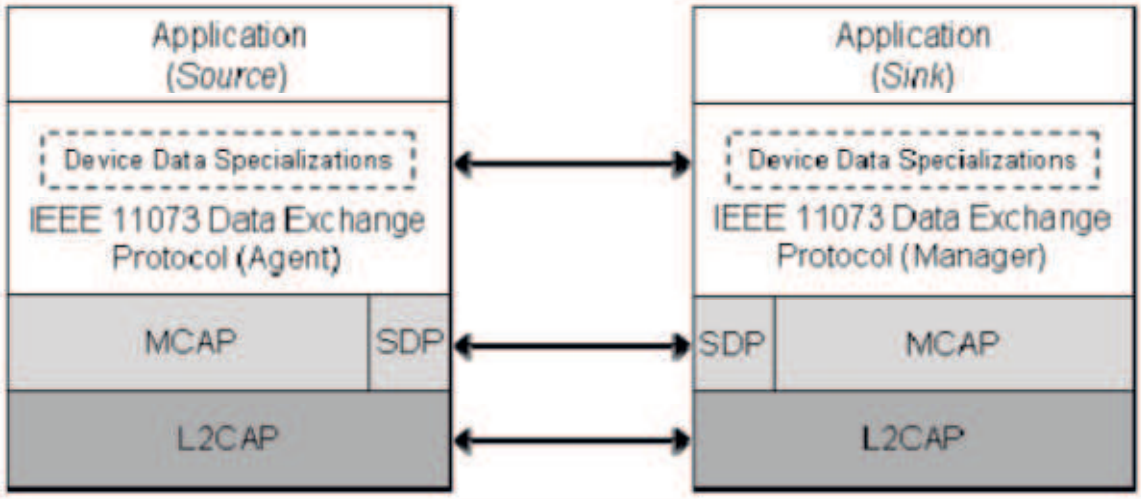
O Bluetooth utiliza radio frequência para efetuar a identificação e transmissão dos dados para os dispositivos que estão dentro de um determinado perímetro de alcance. Para a troca de informações existem dois modos de operação do Bluetooth, os quais são *master* e o *slave*. O *master* é responsável por estabelecer e coordenar a troca de dados. Já o *slave* é submisso ao *master*, e deve consequentemente atender as requisições enviadas pelo mesmo.

O Bluez é a principal implementação da especificação Bluetooth para sistemas operacionais baseados em Linux, ou seja, toda aplicação que necessita utilizar o Bluetooth de um determinado dispositivo deve fazer uso desta implementação. Alguns exemplos de aplicações que utilizaram o Bluez são o hciconfig, hcitools e bluez-tools. Essa implementação foi fundamental para o desenvolvimento do projeto, pois todos os dispositivos médicos utilizados durante a comunicação utilizavam o Bluetooth como meio de comunicação. Além disso, o sistema operacional do Health Aggregator Manager foi construído utilizando o Yocto Project, que é uma ferramenta de construção de sistemas operacionais baseados no Linux.

2.2 Bluetooth Health Device Profile (HDP)

Os perfis de dispositivos produzem grande impacto durante a troca de dados, pois é através deste que são definidos e gerenciados os canais de comunicação, estrutura, sequência e tamanho dos dados. O SSP (Serial Port Profile) é um dos perfis mais usados, pois possibilita certo nível de interoperabilidade, e isso é decorrente da generalidade existente no mesmo, entretanto tal generalidade gera um baixo desempenho durante a troca de informações em alguns tipos de aplicação. Foi ao observar isto que a *Continua Health Alliance* juntamente com a IEEE desenvolveram o perfil HDP objetivando aperfeiçoar as transferências dos pacotes especificados na norma ISO/IEEE 11073-20601 (SIG BLUETOOTH, 2009).

Figura 2 - Arquitetura HDP



Fonte: SIG BLUETOOTH

O HDP possui uma grande relevância para o estabelecimento da comunicação com os dispositivos médicos pessoais, pois é a partir deste perfil que é possível realizar a identificação do tipo de dispositivo médico, seja este um oxímetro, balança, aferidor de pressão, ou outros. Para realizar isto, como ilustrado na Figura 2, a arquitetura simplificada do HDP utilizada o SDP (Service Discovery Protocol) possibilitando analisar todos os serviços que estão sendo processados em um dispositivo Bluetooth (FARIA, 2015).

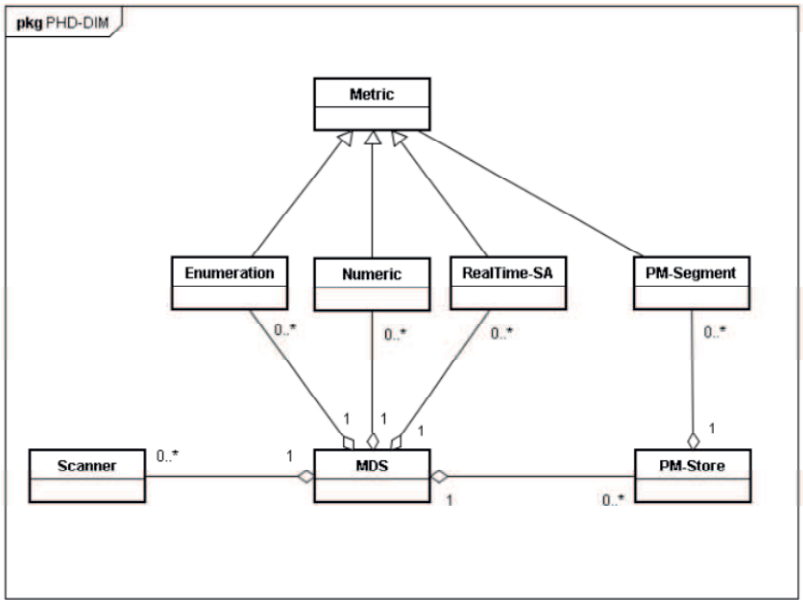
Em decorrência da alta disponibilidade dos dispositivos médicos com a tecnologia Bluetooth juntamente com o HDP implantado nativamente, este trabalho irá adotar este último como protocolo de transporte para o estabelecimento da

comunicação, apesar de ter conhecimento da viabilidade do uso de outros protocolos que dão suporte, como por exemplo, as tecnologias USB e Zigbee.

2.3 Normas ISO/IEEE 11073

As ramificações da norma IEEE 11073 tratam da especificação dos dispositivos médicos pessoais. A principal proposta inerente a estas especificações é o estabelecimento da interoperabilidade. Para alcançar essa meta, estas especificações são fundamentadas no DIM (Domain Information Model) que pode ser visto como a estrutura orientada aos objetos dos componentes básicos comuns a todos os dispositivos que são submetidos a esta norma (ISO/IEEE, 2014). No diagrama UML na Figura 3, é possível ter uma ideia mais concreta da estrutura do DIM. É importante ressaltar que este modelo de informação sempre estará contido dentro do dispositivo médico pessoal.

Figura 3 - Diagrama UML dos componente do DIM



Fonte: Especificação IEEE Std 11073-20601

Uma das classes mais importantes deste modelo é o MDS (Medical Devices System, ou Sistema do Dispositivo Médico). Esta classe possui apenas uma única instância, e em grande parte é composta por diversos outros “objetos filhos”. É nesta classe onde residem todos os atributos relacionados ao dispositivo médico, por exemplo, System ID, informações do fabricante e especialização do dispositivo podendo esta última ser vista como id de identificação associada a cada tipo de

dispositivo médico pessoal, como por exemplo, a balança e o oxímetro que possuem respectivamente as especializações 0x100F, 0x1004.

São através destes atributos que são armazenados os dados obtidos por alguma medição e que devem ser enviados posteriormente para um futuro agregador de dados. Na verdade, o agregador irá se utilizar do MDS para entender a estrutura dos dados de cada dispositivo, para que desta forma possa requisitar os atributos específicos do mesmo (SIGNOVE, 2012).

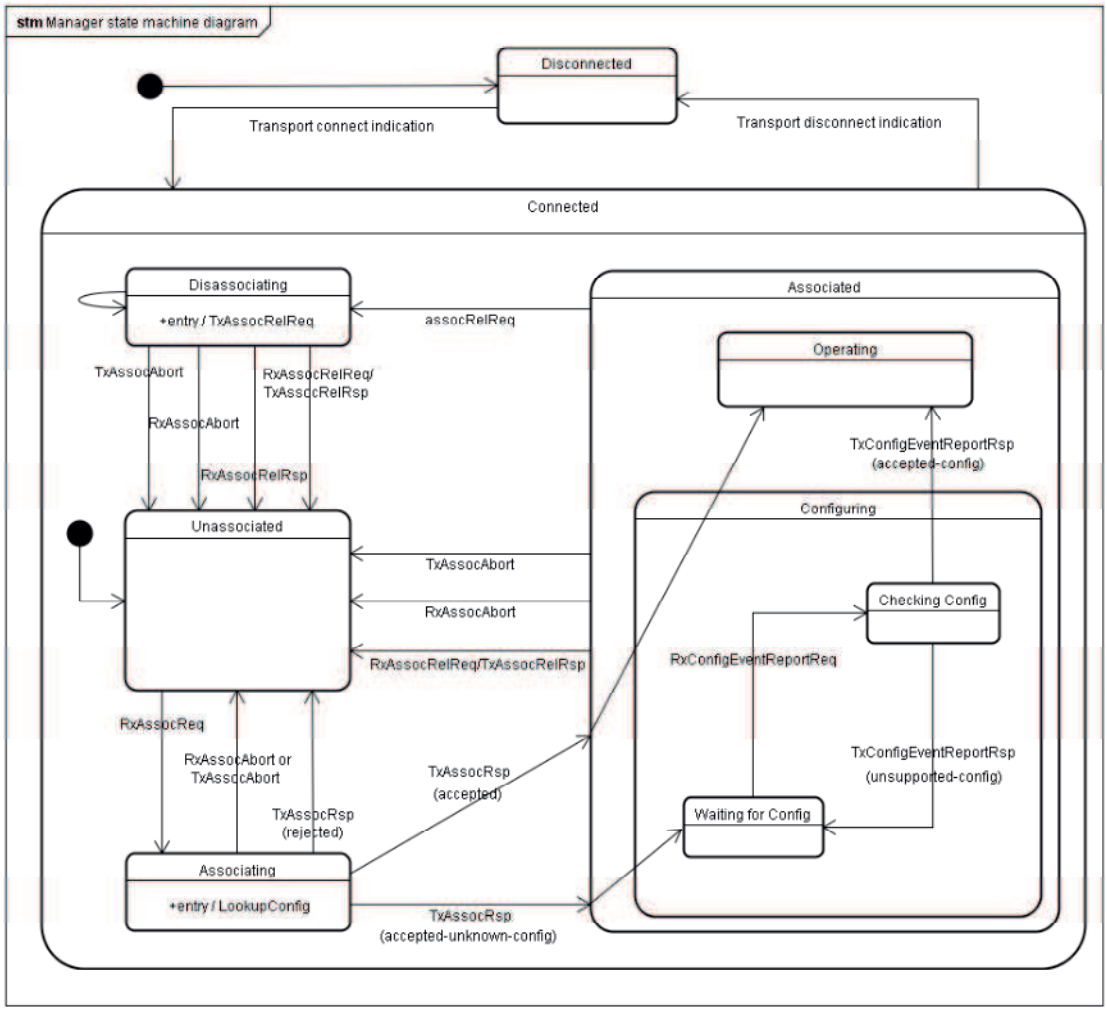
Os outros componentes do modelo DIM juntamente com as suas respectivas descrições são (ISO/IEEE, 2014):

- Metric - classe base abstrata que é herdada pelas classes *Numeric*, *Real-Time Sample Array*, *Enumeration.*
- Numeric - classe utilizada para referenciar uma medida discreta ou uma lista.
- Real-Time Sample Array - utilizado para representar as medidas contínuas.
- Enumeration - descreve informações referentes ao status ou anotações.
- PM-Store - base de dados compostas de diversos PM-Segments.
- PM-Segment - unidade mais básica do PM-Store. É utilizado para o armazenado de medições que posteriormente poderão ser enviadas para o agregador de dados.
- Scanner - monitora os demais componentes e emite eventos notificando um possível agregador de dados quando, por exemplo, o objeto *Numeric* for modificado em decorrência da realização de uma medição.

Para a construção de tal agregador de dados, é utilizado a IEEE 11073-20601 que descreve as interações existentes no mesmo através da máquina de estado na Figura 4.

È possível observar que as transições ocorrem a partir das mensagens emitidas pelo dispositivo médico pessoal, resultando na troca do estado do agregador de dados. Os estados que o mesmo pode assumir são (ISO/IEEE, 2014):

Figura 4 - Máquina de estados de um agregador de dados



Fonte: Especificação IEEE Std 11073-20601

- Desconectado
- Conectado
  - Em fase de associação
  - Associado
    - Esperando por Configuração
    - Verificando Configuração
    - Em Operação
  - Em fase de desassociação

A indentação dos estados mostra a dependência existente entre os mesmos. Um exemplo disso é que é necessário estar Conectado para realizar a Verificação da Configuração.

Apesar de não mais abordar os aspectos técnicos existentes na norma IEEE 11073-20601, todo o trabalho é fundamentado nela, pois são as implementações desta norma que tornam possível realizar o desenvolvimento do Health Aggregator Manager (HAM), uma vez que a mesma é responsável por estabelecer as diretrizes de comunicação dos mais diversos dispositivos médicos pessoais.

Por fim, é relevante mencionar, que a entidade responsável por realizar a validação e a credenciação dos dispositivos médicos desenvolvidos sob esta especificação, é a Continua Health Alliance, garantido assim a interoperabilidade com outros dispositivos que também contenham o "*Continua certified*".

2.4 Antidote

O Antidote é uma implementação da especificação ISO/IEEE 11073-20601 na linguagem de programação C, desenvolvida pela empresa Signove. Assim como na norma em geral, o antidote é fundamentado em dois tipos de dispositivos: os gerentes e os agentes. O gerente é definido como o coletor de dados (o HAM), e podem ser implantados em smartphones, tablets e outros dispositivos embarcados. Em geral, o gerente é tipicamente um poderoso dispositivo que tem a capacidade de conectar-se com a rede e, assim, possibilitar que as informações possam ser compartilhadas para um determinado propósito. Já os agentes podem ser vistos como os dispositivos médicos pessoais, geralmente responsáveis por realizar a aquisição dos dados e, devido a isso, estes são conhecidos como produtores. Apesar de ser possível realizar simulações de dispositivos médicos, o foco do antidote está bem mais relacionado à implementação de um Manager, uma vez que, a maioria das rotinas disponíveis são para este propósito (SIGNOVE, 2012).

O Antidote em si é bastante portável e isso muito se deve a fatores, relacionados a não dependência de frameworks. Outro fator bastante importante que colabora com esta característica é a sua própria arquitetura, que semelhantemente como o Eclipse, é baseado em plug-ins, neste caso plug-ins de comunicação, como por exemplo, USB, Bluetooth, TCP/IP, etc. A seguir algumas das principais funções exercidas pelos plug-ins (SIGNOVE, 2012):

- Notificar quando uma comunicação é aberta e fechada;
- Criar e manter IDs exclusivos para cada dispositivo/conexão;
- Terminar uma conexão quando requisitado;

- Notifica o Antidote quando algum tipo de comunicação chega;

Quando comparado com outras implementações, o antidote é tido como referência em consequência de três características fundamentais. A primeira está associada à disponibilidade do código fonte, uma vez que empresa desenvolvedora estabeleceu a biblioteca sendo OpenSource. A segunda característica está relacionada à possibilidade de se realizar um porte para o android, pois apesar do antidote ser desenvolvido utilizando a linguagem C, ainda assim é possível realizar a comunicação das rotinas do antidote com a máquina virtual do Java, que é amplamente utilizado durante o desenvolvimento de aplicações para o Android através da ferramenta Java Native Interface (JNI). A terceira e última característica está associada à implementação citada que dá suporte a plug-ins de transcodificação. Ou seja, há a possibilidade de dispositivos que foram submetidos à fabricação de um protocolo proprietário de se comunicar com o Antidote.

2.5 OpenHealth Manager

Assim como o Antidote, o OpenHealth Manager é uma outra implementação da norma IEEE 11073-20601 desenvolvida pela *libresoft*. Entretanto, existem algumas características que diferem entre as duas implementações, tais como:

- Esta implementação utiliza a linguagem de programação Java, facilitando a sua adaptação aos mais diversos sistemas, tal como o Android e sistemas embarcados Linux (FARIA, 2015);
- Diferentemente do antidote, esta implementação não possibilita a utilização de transcode, técnica essa utilizada pelo o Antidote para tornar possível a inserção de dispositivos médicos pessoais que não foram fabricados utilizando a respectiva ramificação da norma IEEE 11073, referente à especialização do dispositivo (FARIA, 2015);

Apesar de algumas virtudes oferecidas por esta implementação quando comparada com o Antidote, aparentemente ela está descontinuada, pois o último commit foi realizado na data de 24 de Janeiro de 2011 e o site oficial não está disseminando informações referentes à implementação.

2.6 Comunicação entre Processos

A comunicação entre processos é algo relativamente bem fundamentado, pois é devido a esta comunicação que é possível realizar a fragmentação de sistemas de alta complexidade. Segundo (TANENBAUM, 2009):

> Frequentemente processos precisam se comunicar com outros. Por exemplo, em um pipeline do interpretador de comandos, a saída do primeiro processo deve ser passada para o segundo processo, e isso continua até o fim da linha de comando. Assim, há uma necessidade de comunicação entre processos que deve ocorrer, de preferência, de uma maneira bem estruturada e sem interrupções.

Durante a comunicação de processos estão envolvidos diversos fatores que podem interferir na execução de algum programa como um todo, entre os quais podemos citar:

- Condições de corrida – Estas são caracterizadas quando mais de um processo necessita de recursos compartilhados, afetando o processamento, dependendo da ordem de execução de cada processo;
- Sincronização de Processos – Mecanismo utilizado para realizar o compartilhamento de recursos em regiões críticas. A partir deste é possível ter êxito ao fazer um escalonamento entre os processos, além de evitar a ocorrência de algum deadlock;

Na prática, esta comunicação é realizada através de mecanismos de comunicação entre processos, estes se caracterizam por não ser acoplados a nenhuma linguagem de programação ou framework especifico. Resumidamente, para utilização destes mecanismos o único pré-requisito exigido, é que a linguagem de programação ou o framework tenha a sua disposição alguma biblioteca ou protocolo implementado capaz de estabelecer a comunicação com estes mecanismos. Alguns exemplos destes são:

- Socket – É o mecanismo mais conhecido. É frequentemente utilizado durante a implementação de aplicações que fazem uso da arquitetura cliente/servidor, durante uma requisição para acessar um site na internet, por exemplo (ORACLE, 2017);
- MPI – É um padrão de comunicação destinado a sistemas distribuídos. Estes sistemas necessitam de programação com alto desempenho englobado a

computação paralela, através da utilização de vários núcleos de processamento ou até mesmo clusters (BARNEY e LABORATORY, 2015);

- D-Bus - É um sistema de tráfego de mensagens, com o intuito apenas de possibilitar que os processos troquem informações entre si. Além disso, o D-Bus pode gerenciar o clico de vida dos processos, tornando a manipulação simples e confiável (FREEDESKTOP, 2016);

Apesar do conhecimento da existência destes e outros mecanismo, este trabalho destinou-se a optar pela utilização do D-Bus, sob justifica de que o núcleo deste projeto utiliza o sistema operacional Linux, que por sua vez tem o D-Bus como o principal mecanismo de comunicação.

3 METODOLOGIA

Este capítulo é ramificado em duas seções base. A primeira seção trata das ferramentas fundamentais para o desenvolvimento deste trabalho de forma breve, sendo abordadas de forma geral as principais peculiaridades das tecnologias. A segunda ramificação destina-se a mostrar como foram utilizadas tais tecnologias para o desenvolvimento e concretização dos objetivos apresentados anteriormente, além de mostrar as metodologias utilizadas para criação da especificação do agregador de dados para dispositivos médicos pessoais.

3.1 Ferramentas Tecnológicas

Neste tópico serão brevemente apresentadas três tecnologias fundamentais, que podem ser vistas como os alicerces do projeto como um todo. Para cada uma destas tecnologias serão mencionados os principais conceitos e suas estruturas pertinentes às mesmas.

3.1.1 Yocto Project

O Yocto Project é uma ferramenta utilizada para criação de sistemas operacionais personalizados baseados no Linux. Segundo o grupo da Fundação Linux (LINUX FOUNDATION, 2015):

> O Yocto Project disponibiliza uma infraestrutura e ferramentas para ajudar os desenvolvedores a criar suas próprias distribuições personalizadas Linux para qualquer arquitetura de hardware. O Yocto Project destina-se a fornecer um ponto de partida útil para desenvolvedores.

Em termos gerais o Yocto Project é a junção de diversos outros projetos, cada um destes com uma função específica no processo de compilação de uma imagem de sistema operacional, alguns desses projetos são os seguintes (EMBEDDED LABWORKS, 2014):

- Openembedded Core
- BitBake - Ferramenta escrita em Python responsável por processar e executar as receitas;
- Poky - Sistema de compilação utilizado no Yocto Project;

- AutoBuilder - permite integrar e testar as mudanças realizadas por diferentes desenvolvedores de um projeto utilizando o Yocto Project;
- Hob - O Hob é uma interface gráfica para o BitBake, visando facilitar a configuração e geração de uma imagem Linux customizada.
- Toaster - O Toaster, originalmente chamado de Web Hob, é uma interface Web para o BitBake. Liberado a partir do release 1.6 "Daisy" do Yocto Project, permite configurar e compilar uma imagem, além de analisar informações do processo de build através de uma interface web.

O pleno entendimento e funcionamento do Yocto consiste na implementação de conceitos relacionados à sua arquitetura, tais conceitos são estes:

- **Tarefa (task):** Etapa executada pelo sistema de compilação (fetch, patch, configure, compile, package, etc).
- **Receita (recipe):** conjunto de tarefas para compilar determinado software (.bb , .bbappend).
- **Classes (classes):** herança e encapsulamento da lógica para a execução de tarefas comuns entre as receitas (.bbclass).
- **Pacote (package):** resultado do processamento da receita de um componente de software, agregado em algum formato popular de empacotamento (.ipk , .deb , .rpm).

As principais vantagens fornecidas pelo Yocto estão relacionadas à sua arquitetura, a qual é baseada em camadas. É desta forma que essa ferramenta oferece modularidades e facilidades em manter e estender o sistema através da implementação destas camadas. Outro fator que põe o Yocto em evidência é a grande comunidade que é sempre ativa para sugerir diversas soluções para os possíveis questionamentos e dúvidas que venham a surgir.

Os principais fatores que colaboraram para a adoção dessa tecnologia residem na possibilidade favorável de integrar apenas à imagem do sistema os módulos e os programas que serão realmente necessários, sendo possível descartar os que não são intrinsecamente necessários para atingir os objetivos referentes ao agregador de dados. Como consequência disso, o sistema terá um maior desempenho durante a execução das aplicações.

3.1.2 Node.js

A linguagem de programação *JavaScript* está se disseminando pelas mais variados locais de desenvolvimento de aplicações. Em decorrência disso, diversas plataformas e frameworks foram, e estão sendo, criados para auxiliar a implementação de programas usando esta tecnologia.

O Node.js é uma plataforma de execução de código JavaScript que conquistou de forma consolidada uma vasta comunidade. As principais características referentes à arquitetura são as seguintes:

- Assincronicidade – Uma das vantagens relacionadas ao assincronismo é a otimização em operações de processamento de I/O, pois quando requisitados por milhares de pessoas operações deste tipo, em um servidor, podem necessitar de um considerável período de tempo. As principais características associadas ao assincronismo presente no Node.js são decorrentes do mecanismo EventLoop, responsável pelo monitoramento dos eventos acionados. Em termos gerais, o EventLoop é um loop infinito que constantemente verifica se algum evento foi disparado. Para que isto ocorra, todos os módulos do Node.js herdam o módulo EventEmitter, responsável por gerenciar a emissão de eventos.
- Gerenciamento de módulos NPM (Node Package Manager) – Esta ferramenta foi integrada ao Node.js devido a dois principais aspectos que otimizam o desenvolvimento de software, além de estimular e facilitar o reuso de componentes fornecidos por terceiros. Em primeiro lugar, o NPM funciona como um repositório online de código aberto, no qual componentes podem ser compartilhados e reusados. Em segundo lugar, questões relacionadas a fácil implantação e gerenciamento de versão, favorecem a adesão do NPM, pois possibilita ainda o gerenciamento das dependências inerentes a cada componente.

3.1.3 D-Bus

A estrutura do D-Bus é constituída a partir do entendimento de quatro conceitos para que seja possível estabelecer a comunicação entre processos ativos. O primeiro é o tipo de conexão e este pode ser definido através de dois modos. O primeiro modo de sessão é destinado para integração dos aplicativos de uma

sessão de login de usuário ativo, em geral estes processos são iniciados e postos em execução através do próprio usuário. Já o segundo modo de sistema é único e independente de sessão, ou seja, está disponível para todo o sistema operacional e seus processos ativos, fornecendo serviços do sistema operacional.

O segundo conceito está relacionado ao nome do registro, que é definido como um identificador único onde será possível disponibilizar os sinais e métodos de um determinado objeto por um processo ativo. Tais métodos associados a este registro poderão ser requisitados por outros processos, para o processamento de uma operação específica. Como podemos visualizar na Figura 5, o padrão destinado ao nome do registro segue normalmente a estrutura de uma sequência de letras e dígitos separados por pontos. Em geral, o nome do registro está associado ao nome do domínio ou organização que definiu o nome do registro.

O terceiro conceito está relacionado ao caminho do objeto, que possui como objetivo a organização de forma hierárquica, assim como em um sistema de arquivos, dos muitos objetos que um determinado processo pode desejar exportar. Analogamente aos serviços Web, o caminho de objeto poderia ser considerado como parte da URL, assim como demonstrado na Figura 5. Os caminhos de objeto podem ser constituídos de letras, números e caracteres sublinhados. Eles são separados por barras e devem começar com uma barra no início, entretanto, não devem ser finalizados com uma.

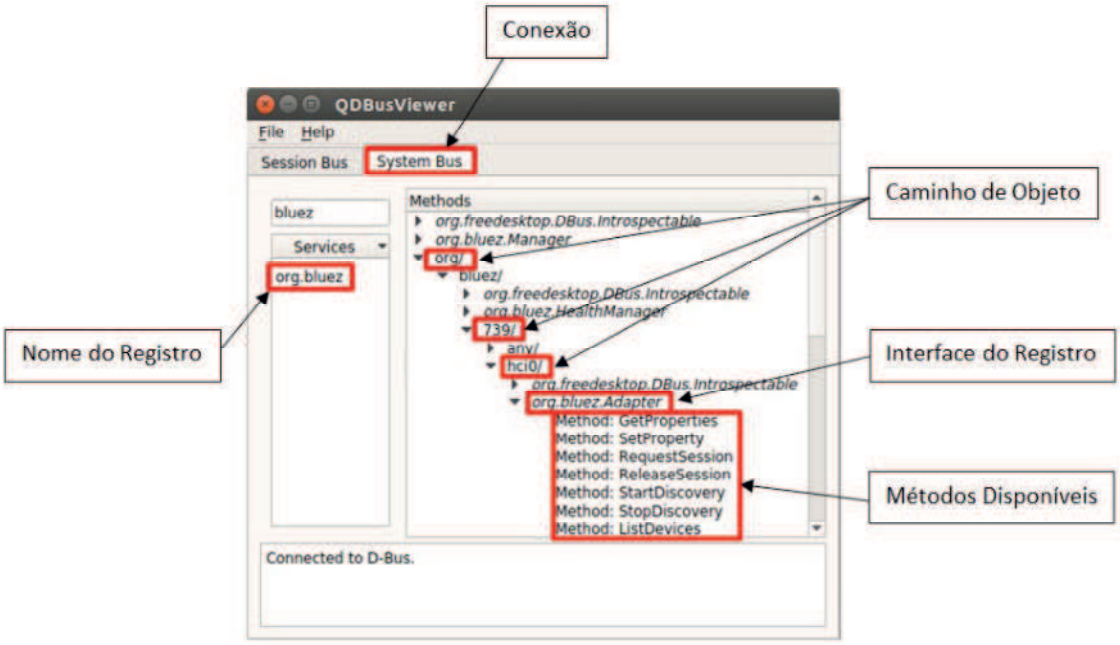
O quarto conceito é definido como a interface do objeto, assim como em uma linguagem de programação orientada a objetos, a interface é basicamente o “contrato” entre o chamador e o destinatário, que consiste da disponibilização dos métodos, sinais e propriedades de um determinado processo para o mecanismo de comunicação D-Bus. Semelhantemente ao nome atribuído a um registro, o padrão destinado ao nome da interface do objeto segue normalmente a estrutura de uma sequência de letras e dígitos separados por pontos.

Um fator importante a ser mencionado é que todos esses conceitos em sua prática são implementados em forma de cascata, ou seja, ao considerar um cenário no qual se tem como objetivo exportar alguns métodos para o D-Bus, deve-se definir em primeiro lugar o tipo de conexão (de sistema ou de sessão), a partir de então estabelecer o nome do registro, a este é atribuído o caminho do objeto e, por fim, associar a este caminho de objeto as interfaces de objetos que contém as operações que se deseja exportar. Outro possível cenário seria a utilização de

métodos, sinais e propriedades já disponíveis no D-Bus, para isto necessita-se apenas ter conhecimento dos conceitos mencionados anteriormente, e assim seria possível realizar requisições nas diversas operações oferecidas pelos registros disponibilizados através do D-Bus.

Durante o desenvolvimento do projeto foi utilizado a ferramenta gráfica Qt D-Bus Viewer para realizar o monitoramento dos processos inclusos no D-Bus. Na Figura 5 é possível visualizar todos os componentes que constituem o D-Bus.

Figura 5 - Ilustração do Qt D-Bus Viewer



Fonte: Elaborada pelo autor

3.2 Processo de Desenvolvimento

O desenvolvimento de um software, em geral, é guiado através de um processo que é delimitado em quatro fases, as quais são a Especificação, o Projeto e Implementação do Software, a Validação e a Evolução do Software. No qual foi utilizado a metodologia ágil Scrum adaptada para o desenvolvimento do software, uma vez que a mesma possibilita uma maior flexibilidade para acoplar possíveis mudanças que emergiram durante o desenvolvimento.

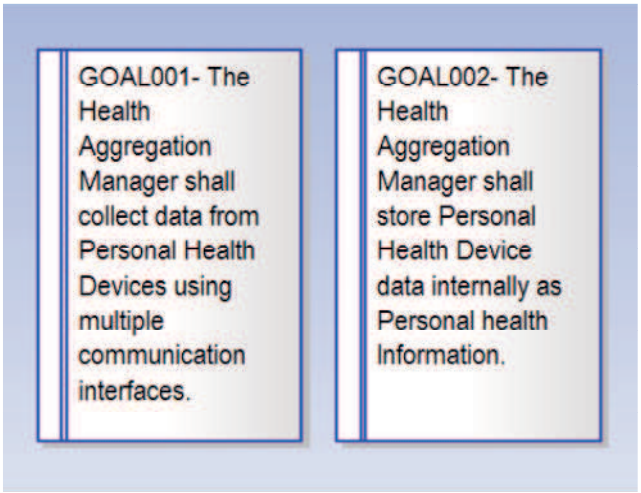
Nesta seção serão observadas as três primeiras etapas desse processo de desenvolvimento, que possuem como funções encontrar os requisitos e restrições do software, projetar e implementar o sistema e, por fim, realizar uma validação do sistema desenvolvido.

3.2.1 Levantamento de Requisitos do HAM

Inicialmente, esse projeto teve como um dos principais propulsores para o seu desenvolvimento a empresa desenvolvedora do Antidote, a Signove. Esta com o intuito de realizar transferência de tecnologia com o Núcleo de Tecnologias Estratégicas em Saúde residente na Universidade Estadual da Paraíba realizou diversas reuniões para a definição dos requisitos a serem implementados. Como resultado disso, diversos artefatos foram gerados, tais como diagramas de casos de uso, possíveis requisitos e o Product Backlog que possui as funcionalidades inerentes ao agregador de dados médicos (observe o Apêndice A).

O Product Backlog é constituído basicamente de quatro informações principais. A primeira delas fornece uma breve descrição a respeito da funcionalidade que se deseja encontrar no agregador de dados. A segunda e terceira informação são designadas a possibilitar a rastreabilidade aos casos de uso e aos requisitos, respectivamente. Por fim, são descritos os critérios de aceitação de determinada funcionalidade que se espera encontrar implementada.

Figura 6 - Definição dos Objetivos



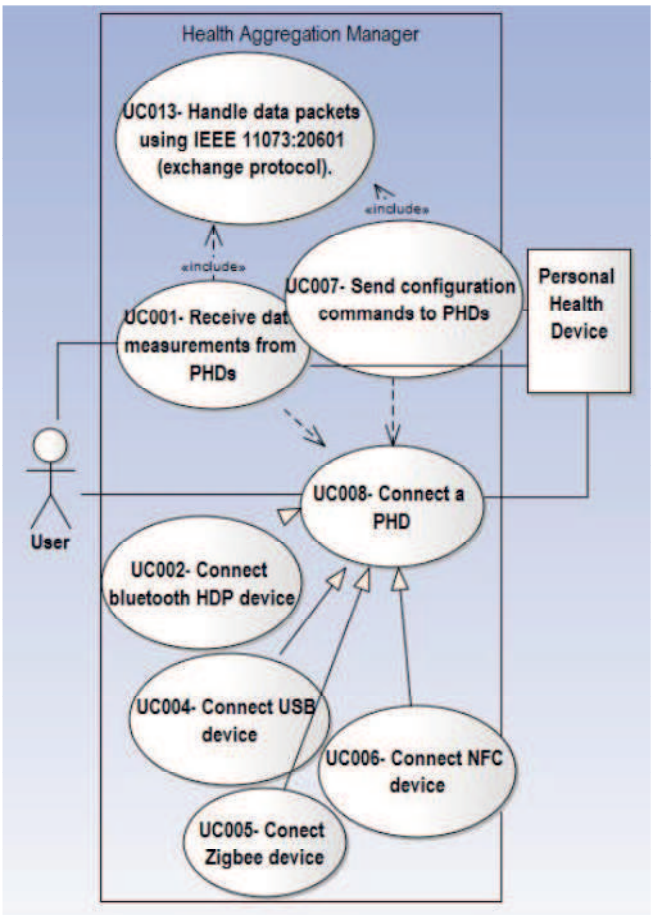
Fonte: Elaborada pelo autor

A partir do Product Backlog foram formalizados os objetivos (Figura 6) juntamente com os casos de uso que expressam os principais componentes e cenários de uso do agregador de dados. O primeiro objetivo está relacionado ao estabelecimento da comunicação entre o gerenciador do agregador de dados e os dispositivos médicos pessoais, tal como a transmissão dos dados médicos obtidos de um determinado paciente. Já o segundo objetivo trata-se da persistência dos

dados, pois uma vez recebidos os dados de um determinado dispositivo médico, deve-se realizar o armazenamento do mesmo em banco de dados, desta forma será possível emitir as informações dos dados vitais de um paciente para um prontuário eletrônico médico, responsável por gerenciar o histórico de saúde de um paciente por exemplo.

Assim como é sugerido em um processo de desenvolvimento ágil, apenas os principais casos de uso foram especificados. Como é possível observamos na Figura 7, os principais atores a interagirem com o agregador de dados são os dispositivos médicos pessoais e um usuário (paciente, médico, etc.). Para o estabelecimento da comunicação e a realização das operações com os dispositivos médicos é utilizado algum plug-in de comunicação, como por exemplo, USB, Zigbee, NFC. É através desses plug-ins e de uma implementação do protocolo IEEE 11073:2061 que é possível realizar o recebimento dos dados provindos.

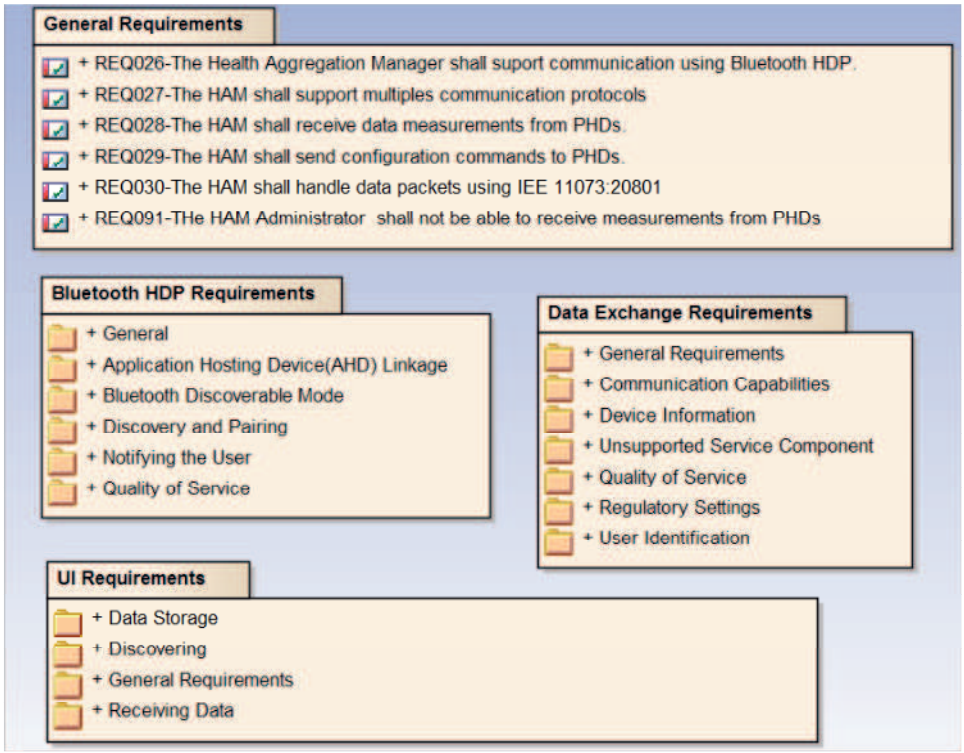
Figura 7 - Diagrama dos casos de uso do HAM



Fonte: Elaborada pelo autor

Por fim, foi realizado o levantamento dos requisitos a serem desenvolvidos juntamente com suas respectivas restrições. A Figura 8 demonstra o panorama geral dos requisitos do agregador de dados. Os requisitos foram divididos em quatro categorias, são estas:

Figura 8 - Diagrama com os requisitos do HAM



Fonte: Elaborada pelo autor

- *General Requirements* – Especifica de forma abrangente as principais tecnologias que o agregador de dados deve estar apto a suportar, tais como Bluetooth HDP (Health Device Profile);
- *Bluetooth HDP Requirements* – Grupo de requisitos relacionados ao estabelecimento e manutenção da conexão com os dispositivos médicos Exemplos de requisitos funcionais contidos nesta categoria são a descoberta de dispositivos com capacidade de comunicação através do Bluetooth, sejam estes dispositivos médicos ou não. Além do pareamento e desparemento com tais dispositivos;
- *Data Exchange Requirements* – Requisitos relacionados às informações que se deseja obter dos dispositivos médicos conectados, tais como sua especialização (Identificação do dispositivo médico), nome do fabricante,

modelo do dispositivo, e, claro, os dados médicos obtidos através do dispositivo de um paciente em especifico;

- *UI Requirements* – Requisitos referentes à usabilidade, telas de operação e notificação para que o operador do agregador de dados esteja ciente do seu status a todo o momento;

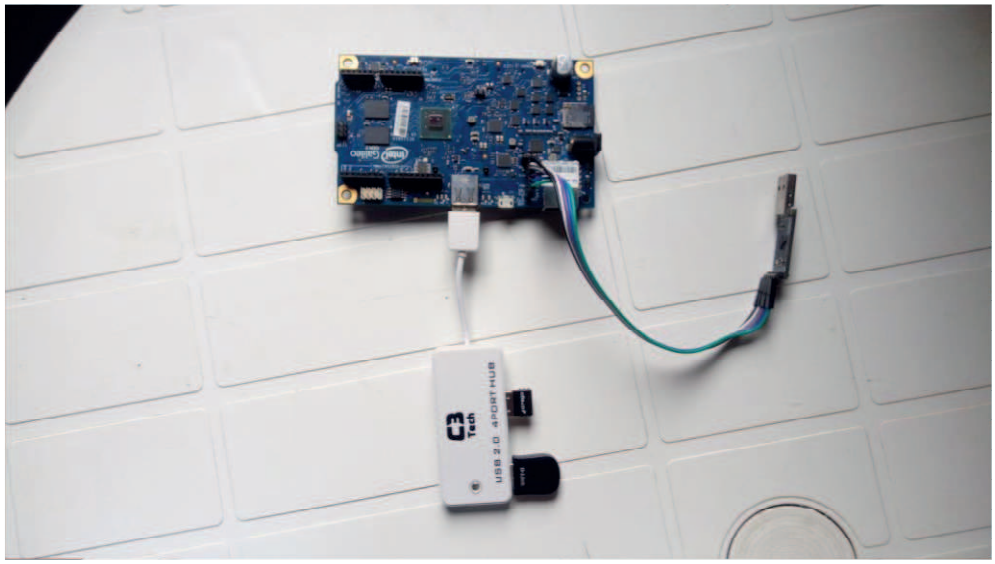
3.2.2 Implementação do HAM

Para o desenvolvimento do agregador de dados médicos foram utilizadas múltiplas ferramentas juntamente com diversas tecnologias. Os componentes físicos utilizados para o desenvolvimento da aplicação são:

- Galileo Gen2, Adaptador Bluetooth 3.0 e Wifi (Figura 9);
- Dispositivo médico (Figura 10);

Uma vez tendo este material e a especificação do sistema à disposição, foram estabelecidos as seguintes etapas para o desenvolvimento do software, estas consistiram na definição do sistema a ser implantado na galileo com os módulos necessários, no desenvolvimento de um servidor para o gerenciamento das operações do agregador de dados e com o estabelecimento da comunicação entre o Manager, implementado através do antidote e um servidor Web.

Figura 9 - Galileo Gen2 e Adaptadores



Fonte: Elaborada pelo autor

3.2.2.1 Definição do Sistema Embarcado na Galileo

Neste projeto do agregador de dados, o Yocto foi utilizado no desenvolvimento de uma imagem do sistema operacional personalizado para a plataforma embarcada na qual será implantado. Tal imagem foi construída tendo em mente as exigências para a perfeita implantação da biblioteca antidote, com o qual é possível estabelecer a comunicação entra a Galileo (o manager de forma geral) e os dispositivos médicos pessoais (os agentes). Isto foi possível devido a disponibilidade da Intel em fornecer algumas camadas de configuração exigidas pelo Yocto Project para geração da imagem do sistema operacional para a plataforma embarcada Galilgeo Gen 2.

Figura 10 - Aferidor de Pressão

Fonte: Elaborada pelo autor

Os principais módulos implantados na imagem do sistema operacional foram os seguintes:

- Módulo do Bluez – Implantou-se este módulo para que o sistema estivesse apto a estabelecer a comunicação com os dispositivos médicos pessoais com tecnologia bluetooth, através de um plugin de comunicação pertencente ao antidote. Uma peculiaridade observada durante o desenvolvimento deste projeto referente ao Bluez, foi a exigência por parte do antidote em requisitar de forma obrigatória a implantação de uma versão da implementação superior ao 4.80. Constatou-se que tal exigência ocorria em decorrência da não existência do HDP (Health Device Profile) em versões anteriores,

característica essa de fundamental importância para o antidote conseguir realizar a identificação dos dispositivos médicos pessoais.

- Módulo do Node.js - O projeto como um todo funciona em torno desta plataforma de execução de JavaScript. Ela foi utilizada para desenvolver um servidor na Galileo capaz de gerenciar as operações referentes ao agregador de dados médicos;
- O Módulo do Wifi e Ethernet – Responsável por tornar possível a inserção da nossa plataforma embarcada em uma rede de comunicação. Fator este, que habilita os usuários do HAM a fazer uso das operações através de outros dispositivos que também estão conectados na mesma rede da galileo.
- Módulo D-Bus – Em decorrência da necessidade na comunicação dos principais componentes do sistema do agregador de dados, foi utilizado este mecanismo IPC, para realizar a interação com os diversos processos existentes.
- Módulo SQLite – As informações dos dados transferidos dos dispositivos médicos pessoais para o agregador de dados, isto é a Galileo, foram armazenadas em uma base de dados com o intuito de no futuro tornar possível o envio destes dados para prontuários médicos online, mais conhecidos como Electronic Health Records, capazes de armazenar, processar e manter o histórico dos dados referentes a cada paciente de forma individual.

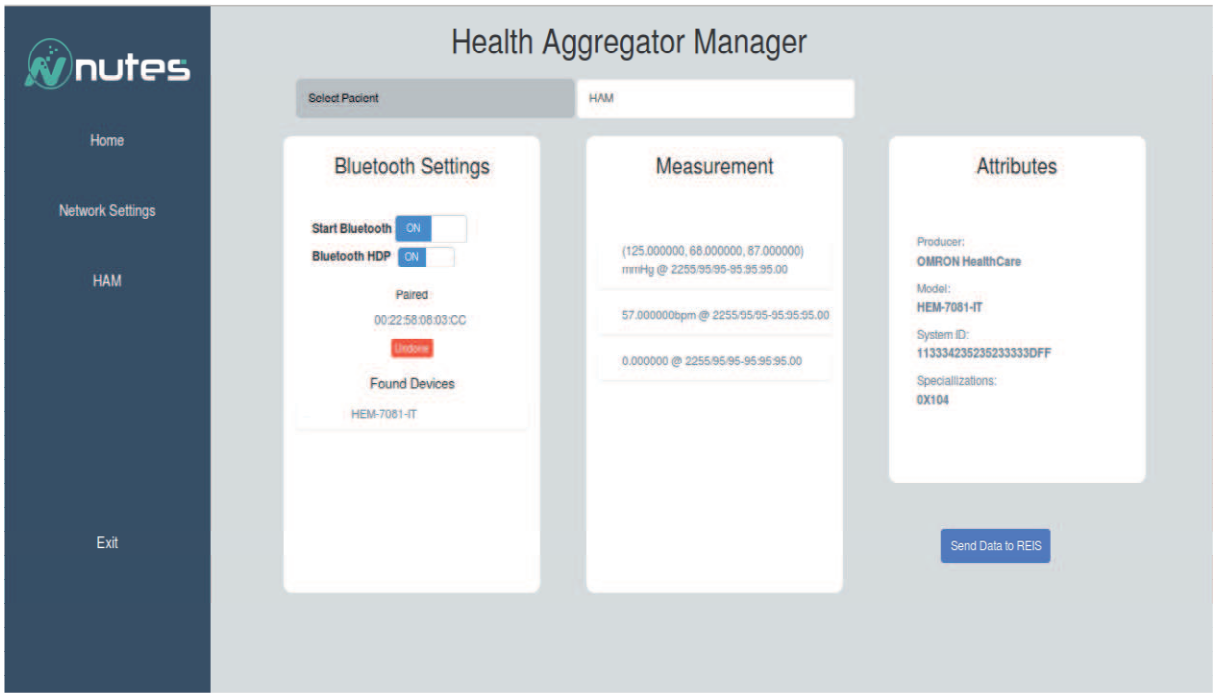
3.2.2.2 Criação do Servidor Web

Para o desenvolvimento do servidor Web que foi implantado no sistema operacional embarcado na Galileo Gen 2, foram utilizadas diversas tecnologias que processaram de forma colaborativa suas funcionalidades especificas para o pleno funcionamento do agregador de dados. O principal objetivo do servidor web é oferecer suporte a uma interface gráfica para manipulação das operações. Para alcançar tais objetivos, foram utilizadas as seguintes tecnologias:

- HTML5 – Foi utilizada a quinta versão do HTML para estruturar as páginas web e seus conteúdos. Os principais fatores que colaboram para a adesão desta versão são decorrentes do suporte à semântica do código e da facilitação na manipulação dos elementos por parte do CSS e JavaScript;
- CSS3 – Bastante popular no desenvolvimento de páginas web na atualidade devido à possibilidade de inserir animações nas mesmas, esta tecnologia foi implantada durante a estilização das páginas web, com o intuído apenas de tornar a interface mais intuitiva para a fácil manipulação do usuário final;
- Vue.js – Este é um framework desenvolvido em JavaScript, utilizado no gerenciamento do frontend das páginas web. Em termos gerais, o Vue.js possibilita criar páginas interativas auxiliando na manipulação dos componentes de uma páginas Web;
- Express.js – Este é um dos principais módulos utilizados no desenvolvimento de aplicações JavaScript que utilizam a plataforma de execução Node.js. O principal motivo para a adesão deste módulo é devido ao mesmo proporcionar uma fácil manipulação do servidor implantado, tornando o desenvolvimento do mesmo rápido e gerenciável. Neste projeto, os principais aspectos explorados deste módulo são referentes às requisições cliente-servidor das páginas web, bem como a criação e administração do servidor.
- Socket.io – Foi utilizado para manter uma conexão em *real-time* com o intuito de monitorar o recebimento de possíveis dados enviados de dispositivos médicos, e direcionar tais dados para a página web. Esta tecnologia também é desenvolvida em JavaScript, e também foi incorporada ao projeto do agregador de dados através da inserção do seu respectivo módulo disponibilizado no NPM.

Por fim, como será mencionado na seção 3.2.2.5 é possível perceber que toda a integração dos módulos são associados ao servidor web, uma vez que este possui as páginas web capazes de redirecionar as requisições para o respectivo módulo. Na Figura 11 é possível visualizar as operações de bluetooth e a recepção dos dados médicos de um determinado dispositivo, juntamente com as informações do modelo dos mesmos. Foi exatamente no desenvolvimento destas e outras páginas web que as tecnologias mencionadas anteriormente foram utilizadas.

Figura 11 - Tela de gerenciamento do HAM



Fonte: Elaborada pelo autor

3.2.2.3 Criação do Módulo Bluetooth

O Módulo bluetooth foi construído para operar em dois modos de pesquisa. O primeiro funciona de forma padrão, sendo possível localizar e parear com os mais diversos dispositivos, como por exemplo, smartphone, tablets ou computadores. Já o segundo modo trabalha de forma mais específica, possibilitando apenas a localização e o pareamento de dispositivos médicos pessoais que estejam em uma curta faixa de distância, no máximo de 10 metros. Para que seja possível realizar a identificação de um dispositivo médico é analisado o perfil do dispositivo(HDP). Este perfil tem como uma de suas principais características a descrição dos serviços inerentes a um dispositivo específico. Esta é uma das maneiras de possibilitar a distinção do bluetooth de um computador, TV, smartphone ou fones de ouvido.

Como em todo o projeto, o JavaScript esteve presente juntamente com o Node.js. No desenvolvimento do módulo Bluetooth não foi diferente, no entanto foram encontrados desafios em decorrência da não existência de nenhum módulo Bluetooth utilizando o BlueZ no NPM. Para solucionar tal empecilho, foi necessário se comunicar com os métodos Bluetooth disponibilizados no D-Bus. A partir disso, foi possível construir os métodos em JavaScript capazes de realizar as operações de pesquisa dos dispositivos (com HDP ou não), pareamento e despareamento. Para o estabelecimento da conexão entre o mecanismo de comunicação e o processo ativo, foi utilizado o módulo “dbus” disponibilizado no NPM por *shouqun e fredchien*. Na Figura 12 é possível observarmos alguns dos principais métodos do BlueZ utilizados para compor a solução do problema.

Com objetivo de obtermos um pleno entendimento do modo de operação do D-Bus, imaginemos, pois, o cenário no qual o *Processo1* disponibiliza uma rotina que será capaz de realizar a pesquisa de dispositivos bluetooth e retornar uma lista contendo o nome do dispositivo com seu respectivo MAC para o chamador da rotina. Já o *Processo2* deve requisitar o método de pesquisa do *Processo1* que reside no D-Bus. Para realizar a requisição de um determinado método, é necessário ter conhecimento do nome do serviço, caminho do objeto, interface do objeto e o protótipo do método que é, em geral, disponibilizado através de uma API (veja a Figura 12). De forma prática o *Processo1* pode ser relacionado ao processo do BlueZ mostrado na Figura 5 e o *Processo2* sendo uma aplicação como o agregador de dados.

Figura 12 - Fragmento da API disponibilizada no D-Bus do BlueZ

```
Adapter hierarchy
=================

Service         org.bluez
Interface       org.bluez.Adapter
Object path     /org/bluez/{hci0,hci1,...}

Methods         dict GetInfo()

                        Returns the properties of the local adapter.

                        Possible errors: org.bluez.Error.NotReady

                string GetAddress()

                        Returns the device address for a given path.

                        Example: "00:11:22:33:44:55"

                        Possible errors: org.bluez.Error.NotReady

                string GetVersion()

                        Returns the version of the Bluetooth chip. This version
                        is compiled from the LMP version. In case of EDR the
                        features attribute must be checked.

                        Example: "Bluetooth 2.0 + EDR"

                        Possible errors: none
```

Fonte: Elaborada pelo autor

3.2.2.4 Criação do Manager

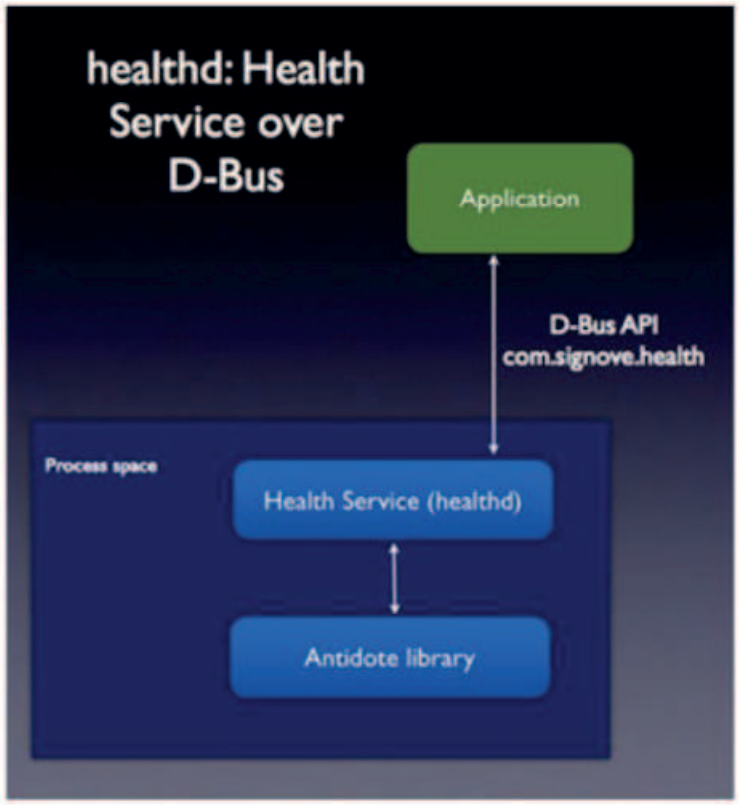
No desenvolvimento do manager foi adotado uma abordagem dirigida a reuso utilizando o componente *healthD*[1]. Este é um componente open source disponibilizado pela Signove, e é destinado a realizar a comunicação entre dispositivos médicos. Este componente deve ser visto como a utilização da Norma IEEE 11073 através do antidote. A integração desse componente com o agregador

[1] Disponível em: https://github.com/signove/antidote. Acesso em Março de 2017

de dados médicos é realizada através de interfaces de comunicação que são disponibilizadas em uma API. Esta utiliza o D-Bus como o mecanismo de comunicação entre processos, para realizar a disseminação das informações obtidas dos dispositivos médicos. Para utilizar tais interfaces é necessário que a camada da aplicação que deseja realizar a comunicação com o healthD (isto é, a interface gráfica do agregador de dados), exporte os métodos exigidos na API para o D-Bus. Desta maneira, sempre que algum dispositivo se conectar, enviar medições, configurações ou atributos, o Manager sempre irá notificar através destes métodos a camada superior a respeito do status da comunicação.

Uma vez que os dispositivos médicos estabelecem a comunicação com o Manager é possível realizar a troca de informações. O Manager também é responsável por encapsular todas estas informações recebidas dos dispositivos médicos em um formato XML e, em sequência, exportá-las para uma camada superior, como é possível analisar na Figura 13.

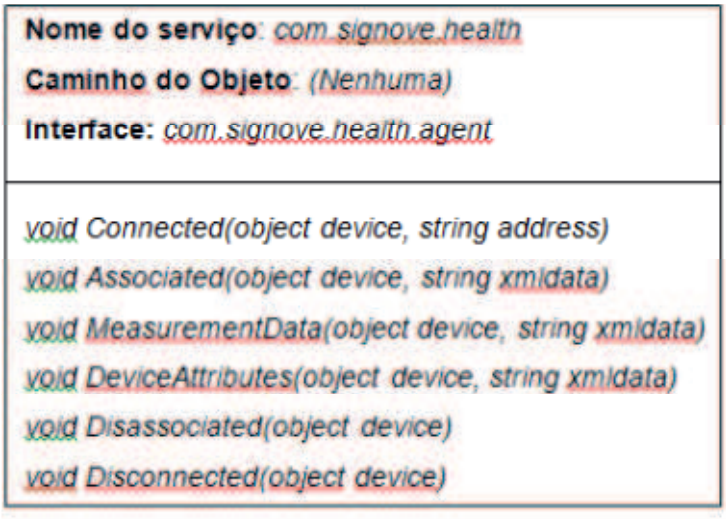
Figura 13 - Arquitetura do healthD



Fonte: Elaborada pela Signove

As principais interfaces de comunicação descritas na API, disponibilizada pela Signove para o estabelecimento da comunicação, podem ser vistas a seguir:

Figura 14 - Interfaces de Comunicação com healthD

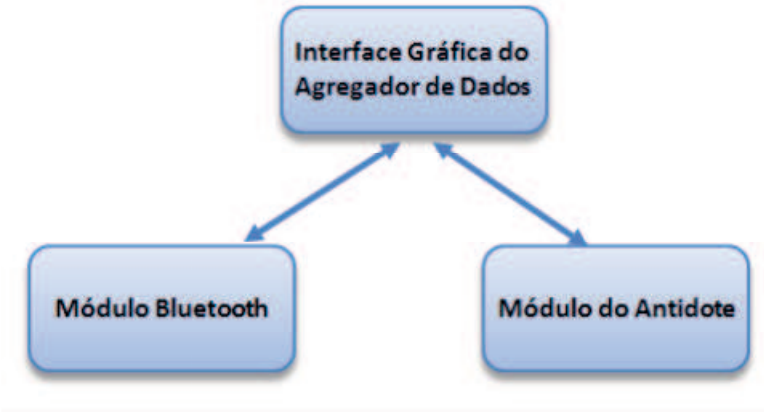


Fonte: Elaborada pelo autor

3.2.2.5 Estabelecendo a Comunicação entre o Manager, o Módulo Bluetooth e o Servidor Web

A partir da criação dos módulos que constituem este projeto, foi necessário utilizar novamente o mecanismo de comunicação D-Bus para que os mesmos fossem capazes de trabalhar de forma colaborativa. Tal objetivo foi atingido através da integração destes componentes com a interface gráfica, como demonstrada na Figura 15.

Figura 15 - Comunicação dos componentes do HAM



Fonte: Elaborada pelo autor

De forma sintetizada, a interface gráfica requisita os métodos necessários para a execução de um determinado procedimento residente no componente Bluetooth (mencionado na seção 3.2.2.3) ou no Antidote, isto é, o *HealthD* (mencionado na seção 3.2.2.4). Com esta integração em um cenário normal, sempre que for necessário realizar alguma operação referente ao Bluetooth, como por exemplo, pesquisa de dispositivos normais ou médicos, pareamento ou desparemento, tais operações serão redirecionadas do módulo da interface gráfica para o módulo Bluetooth. De forma semelhante, ocorrerá com o módulo do *Antidote*, exceto pelo fato de que, são as informações provenientes dos dispositivos médicos pessoais que serão redirecionados a interface gráfica do agregador de dados.

4 CONCLUSÃO

Neste trabalho foi desenvolvido um agregador de dados médicos que está em conformidade com a norma IEEE 11073-20601, utilizando a biblioteca Antidote desenvolvida pela empresa Signove. Do ponto de vista contributivo, foi demonstrado a possibilidade de gerenciar diversos dispositivos médicos pessoais, independentes de seus fabricantes, desde que estes implementem a norma ou ao menos disponibilizem o protocolo proprietário utilizado na fabricação dos dispositivo. Desta maneira se torna possível realizar a construção de um plugin de transcodificação seguindo as exigências do Antidote.

A partir deste trabalho acredita-se ser possível analisar diversas ramificações que podem ser investigados. Alguns destes possíveis trabalhos podem estar relacionados à análise dos dados médicos residentes no agregador de dados, obtidos dos dispositivos médicos. Tal projeto poderia utilizar o Big Data como base do desenvolvimento. Outra possibilidade seria o gerenciamento das informações de um determinado paciente através de um prontuário eletrônico médico capaz de manter a sincronização com os dados contidos no agregador.

Em decorrência do surgimento do movimento de Internet of Things, novos meios para o estabelecimento da interoperabilidade e conectividade de dispositivos médicos pessoais estão surgindo e podem ser comparados e complementados com sistemas similares à esta solução do agregador de dados médicos. Atualmente, um novo padrão que tem se estabelecido é o Bluetooth GATT, que apesar de ser considerado novo, ainda assim, é fortemente fundamento na norma IEEE 11073. Este padrão vem ganhando uma maior visibilidade devido a este estar acoplado as novas tendência de Bluetooth Smart e Bluetooth Low Energy.

Por fim, todo conhecimento adquirido no desenvolvimento deste trabalho de conclusão de curso visa ser integrado ao contexto da fábrica de dispositivos médicos, que será implantada no futuro na Universidade Estadual da Paraíba, a qual terá o Núcleo de Tecnologias Estratégicas em Saúde (NUTES) como principal supervisor das atividades lá realizadas.

REFERÊNCIAS

ASARE, et al. The medical device dongle: an open-source standards-based platform for interoperable medical device connectivity. **2nd ACM SIGHIT International Health Informatics Symposium**, Miami, 28 30 2012.

BARNEY, ; LABORATORY, L. L. N. Message Passing Interface (MPI). **Livermore Computing Center**, 2015. Disponivel em: <https://computing.llnl.gov/tutorials/mpi/>. Acesso em: 10 Maio 2017.

EMBEDDED LABWORKS. **Embedded Labworks**, 2014. Disponivel em: <https://e-labworks.com/treinamentos/yocto-project/>. Acesso em: 20 Maio 2017.

FARIA, R. M. C. **Middleware para comunicação com dispositivos médicos**. Faculdade de Ciências da Universidade do Porto. [S.l.]. 2015.

FREEDESKTOP. D-Bus, 2016. Disponivel em: <https://www.freedesktop.org/wiki/Software/dbus/>. Acesso em: 11 Maio 2017.

ISO/IEEE. **11073-20601-2014 - health informatics–personal health device communication - part 20601: Application profile- optimized exchange protocol**. [S.l.]. 2014.

LIMA, A. C. R.; GONÇALVES, G. L. **Transmissão de Áudio sem fio por Tecnologia Bluetooth**. Universidade de Brasília. Brasília. 2009.

LINUX FOUNDATION. **Yocto Project**, 2015. Disponivel em: <https://www.yoctoproject.org/>. Acesso em: 11 Maio 2017.

ORACLE. The Java™ Tutorials, 2017. Disponivel em: <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>. Acesso em: 20 Março 2017.

ROUSE, M. IoMT (Internet of Medical Things) or healthcare IoT. **IoT Agenda**, 2015. Disponivel em: <http://internetofthingsagenda.techtarget.com/definition/IoMT-Internet-of-Medical-Things>. Acesso em: 13 Fevereiro 2017.

SIG BLUETOOTH. **Health device profile: Implementation guidance whitepaper**. [S.l.]. 2009.

SIGNOVE. **Antidote: Program Guide**. Signove. Campina Grande. 2012.

SOUZA, A. S. D. **Especificação de uma Interface de Comunicação entre um Agregador de Dados em Saúde e um Ambiente Clínico Integrado**. Campina Grande: [s.n.], 2016.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 3ª. ed. [S.l.]: Pearson, 2009.

APÊNDICE A – Product Backlog

ID	Epic Theme	User Story	Acceptance Criteria	Sprint
1	Goal001	Personal Health Device Manager: Bluetooth HDP – Discovery and Pairing	**Short Description:** Implement base Bluetooth module to allow/control the discovery and pairing of Bluetooth health devices. It is necessary to offer an API to be used by the UI to access the list of paired/discovered devices. **Reference Use Cases:** - UC008: Use communication Protocol (Partial) - UC002: Use Bluetooth HDP (Partial) **Acceptance Requirements:** Bluetooth HDP*: REQ001 –> REQ025 **Deliverables:** - Module with API allowing: Discovery, Choosing and Pairing of devices. - API documentation.	#1
6	Goal001	User Interface: Discovery and Register Device	**Short Description:** Simple UI allowing the discovery of PHD. This UI must be designed to work in full screen mode. **Reference Use Cases:** -UC002: Use Bluetooth HDP (Partial) **Pre-requirements:** Bluetooth HDP*: REQ001 –> REQ025 **Acceptance Requirements:** NEW REQs for UI **Deliverables:** - Application with simple to use User Interface allowing the discovery and pairing of	**#1**

			Bluetooth HDP devices in-app. (not using system apps). - Simple manual documentation.	
2	Goal001	Personal Health Device Manager: Bluetooth HDP – Connect and Disconnect	**Short Description:** Integrate in the Bluetooth module of the Personal Health Device Manager the feature to connect and disconnect from a PHD. This module must send notifications by an API to upper layers (such as the UI) indicating that a device is connected or not. **Reference Use Cases:** - UC008: Use communication Protocol (Partial) - UC002: Use Bluetooth HDP (Partial) - **Acceptance Requirements:** N/A **Deliverables:** - Bluetooth Module of the Personal Health Device manager indicating that a device is connected or not. - API Documentation updated.	
3	Goal001	Personal Health Device Manager: Handle data packets using IEEE 11073	**Short Description:** The PHDM must support IEEE 11073 devices. The PHDM must use Antidote IEEE 11073 library, and must be integrated with extra Bluetooth Modules. **Reference Use Cases:** - UC013: Handle data packets using IEEE 11073:20601 (exchange protocol). **Acceptance Requirements:** REQ080 **Deliverables:** - PHDM using Bluetooth module for discovery/pairing, integrated with Antidote	

			Module.	
4	Goal001	Personal Health Device Manager: Bluetooth HDP – Receive Data	**Short Description:** Full integration of Bluetooth features with Antidote. The PHDM must offer an API for discovery/pairing of devices, and must be able to notify upper layers (UI) about the receiving and connection status. **Reference Use Cases:** - UC008: Use communication Protocol (Partial) - UC002: Use Bluetooth HDP (Partial) - UC001: Receive data measurements from Personal Health Devices (Partial) - UC013: Handle data packets using IEEE 11073:20601 (exchange protocol). **Acceptance Requirements:** REQ023, REQ028 **Deliverables:** - PHDM module running with a holder application receiving data from oximeter devices. - API documentation updated.	
5	Goal002	Personal Health Device Manager: Bluetooth HDP – Save Data	**Short Description:** The PHDM must store the health data in an internal database, to be accessed by external components. **Reference Use Cases:** - UC008: Use communication Protocol (Partial) - UC002: Use Bluetooth HDP (Partial) - UC001: Receive data measurements from Personal Health Devices (Partial) - UC013: Handle data packets using IEEE 11073:20601 (exchange protocol). **Acceptance Requirements:**	

			N/A **Deliverables:** - PHDM module integrated with Bluetooth saving IEEE 11073 data packets in internal DB.
7	Goal001	User Interface: Show Connection Status	**Short Description:** The application UI must have a notification area showing the current status of a connection (connecting, connected, disconnecting, disconnected). **Reference Use Cases:** -UC002: Use Bluetooth HDP (Partial) - ??? **Acceptance Requirements:** **N/A** **Deliverables:** - Application showing the status of current connections in a notification area.
8	Goal001	User Interface: Show Received Data	**Short Description:** The application UI must exhibit measurements received by the PHDM in the area. Must show the value, unit and timestamp **Reference Use Cases:** - UC001 – Receive data measurements from Personal Health devices. **Acceptance Requirements:** REQ023 ?? **Deliverables:** - Application exhibiting received measurements.
	Goal002	User Interface: Show Saved Data	**Short Description:** The UI must have a section where is showed the current saved data of the PHDM as a list.

			Reference Use Cases: ?? **Acceptance Requirements:** N/A **Deliverables:** - Application showing stored data of the HAM as a list.	
		Template:template	**Short Description:** **Reference Use Cases:** **Acceptance Requirements:** **Deliverables:**	